# Методичка 9.1 - Поиск бинарных уязвимостей

**Примеры бинарных уязвимостей**

*Переполнение буфера*
*Целочисленные переполнения*
*Уязвимости формата строки*

Более подробно с бинарными уязвимостями вы познакомитесь в следующем курсе, который как раз им посвящен, а сейчас мы просто перечислим несколько примеров бинарных уязвимостей, чтобы вы понимали о чем идет речь.

Наиболее ярким примером являются уязвимости переполнения буфера, который может располагаться как в стеке, так и в куче. Там есть свои нюансы.

Не менее опасны уязвимости целочисленного переполнения, которая очень часто приводят к нарушению логики работы программы и, как следствие, например, к обходу аутентификации.

Уязвимости формата строки, на первый взгляд, может показаться, не такой опасной, но при определенных обстоятельствах эксплуатация такой уязвимости может привести к исполнению произвольного кода.

**Разница между 0-day и 1-day уязвимостями**

Наверно, многие слышали такой термин, как 0-day уязвимость. Этот термин переводится на русский язык, как уязвимость нулевого дня. Почему нулевого? Ноль означает, количество дней после того, как вышел патч, который закрывает уязвимость.

Получается, что если уязвимость в программе есть, а патча от вендора нет, следовательно пользователи не могут обновить свои программы, значит в этот момент времени программа имеет 0-day уязвимость.

Почему так получается? Ситуации могут быть разные. Наиболее распространенный случай, когда кто-то уязвимость нашел, но вендора о ней не уведомил. Соответственно, он может разработать эксплойт и атаковать уязвимые программы или вовсе продать этот эксплойт кому-то еще. Бывают случаи, когда исследователь публикует информацию об уязвимости в сети Интернет, вместо того, чтобы сначала уведомить об уязвимости вендора и дать ему время на выпуск патча. То время, пока вендор будет выпускать патч, уязвимость все еще будет иметь статус 0-day.

Уязвимости 0-day очень часто используются во вредоносном программном обеспечении. Как правило, кто-то находит уязвимость, затем разрабатывает эксплойт и атакует компьютеры в сети Интернет. При таких атаках зона поражения будет очень большой. В процессе расследования инцидента взлома, примеры малвари, попадают к аналитикам в различные антивирусные

лаборатории, где их подробно изучают. Если малварь эксплуатировала неизвестную ранее уязвимость в какой-то программе, например, интернет-браузер, то информацию об уязвимости отправляют вендору, а те выпускают патч.

Теперь давайте разберемся с 1-day уязвимостями. Почему 1? Цифра один здесь подчеркивает то время, которое проходит между выпуском патча и фактическим обновлением программ у пользователей. Не все программы поддерживают автоматическое обновление, не все пользователи оперативно принимают решение по обновлению, таким образом даже после выпуска патча вендором в сети Интернет остается очень много уязвимых компьютеров.

1-day уязвимости также опасны, как и 0-day, так как несмотря на то, что вендор и тот, кто нашел уязвимость публично не раскрывали детали уязвимости, их можно получить путем анализа патча.

**Способы поиска бинарных уязвимостей**

*Фаззинг - 0-day*
*Анализ исходного кода - 0-day*
*Анализ дизассемблированного кода - 0-day*
*Анализ патчей - 1-day*

Давайте теперь немного затронем тему поиска уязвимостей. Как ищут бинарные уязвимости? На слайде перечислены основные способы поиска уязвимостей, давайте выделим те из них, которые могут быть использованы при реверс инжиниринге.

Фаззинг - техника тестирования программного обеспечения, часто автоматическая или полуавтоматическая, заключающаяся в передаче приложению на вход неправильных, неожиданных или случайных данных и последующий анализ результатов поведения приложения.

Анализ исходного кода может применим только тогда, когда исходный код доступен. Это не наш случай.

Анализ дизассемблированного кода - это как раз тот, чем мы занимались практически в течение всего курса. По сути это и есть реверс-инжиниринг.

Анализ патчей является очень эффективным способом для поиска 1-day уязвимостей. Поскольку, имея патч, путем сравнения двух версий бинарного файла до патча и после патча можно выявить все изменения в файле и понять, что было исправлено. Этот способ нам тоже подходит.

**Причины скрытия исправленных уязвимостей**

Хотелось бы еще отметить тот факт, что вендор не всегда публикует информацию об обнаруженных и исправленных уязвимостях. Это может происходить по разным причинам, например, кто-то уведомил вендора о наличии серьезной уязвимости, вендор принимает репорт и тихо исправляет ее и выпускает патч с каким-нибудь абстрактным описанием, например, повысили стабильность работы программы или улучшили скорость работы. Так вендор поступает, потому что  опасается негативной реакции от клиентов или хочет дать больше времени на процесс обновления программ у клиентов.

Но теперь мы знаем, что, имея патч, можно сразу узнать, что было исправлено на самом деле в программе.